CLAIMS

What is claimed is:

1. A method for managing versions of a plurality of software components on a network, comprising:

detecting a version change to a first software component out of the plurality of the software components; and

automatically identifying a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component.

2. The method as recited in Claim 1, further comprising automatically downloading the second software component from a network device.

3. The method as recited in Claim 2, further comprising storing a copy of the second software component in a cache.

4. The method as recited in Claim 3, further comprising checking version information of the second software component that is stored in the cache to determine whether to perform the downloading step.

5. The method as recited in Claim 1, further comprising:

collecting attributes of the second software component; and

automatically manipulating the second software component according to the attributes.

6. The method as recited in Claim 5, wherein the manipulating further includes:

downloading the second software component as part of performing an instance of the method; and

replacing an existing version of the second software component with the second software component that has been downloaded in the same instance.

7. A method for managing versions of a plurality of software components on a network, comprising:

detecting a version change to a first software component out of the plurality of the software components;

automatically identifying a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component;

collecting attributes of the second software component; and

automatically manipulating the second software component according to the attributes.

8. The method as recited in Claim 7, wherein the manipulating further includes:

downloading the second software component as part of performing an instance of the method; and

replacing an existing version of the second software component with the second software component that has been downloaded in the same instance.

9. The method as recited in Claim 8, further comprising further comprising storing a copy of the second software component in a cache.

10. The method as recited in Claim 9, further comprising checking version information of the second software component that is stored in the cache to determine whether to perform the downloading step.

11. An apparatus for managing versions of a plurality of software components on a network, comprising:

a user interface; and

a processing engine, coupled to the user interface, wherein the processing engine further includes:

an event manager that detects a version change to a first software component out of the plurality of the software components; and

a component manager that in response obtains version information of first software component from a version manager and automatically identifies a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component.

12. The apparatus as recited in claim 11, wherein the component manager automatically downloads the second software component.

13. The apparatus as recited in claim 11, wherein the component manager informs an operator of the apparatus of the second software component via the user interface.

14. The apparatus as recited in Claim 12, wherein the component manager stores a copy of the second software component in a cache.

15. The apparatus as recited in Claim 14, wherein the component manager checks version information of the second software component that is stored in the cache to determine whether to download the second software component.

16. The apparatus as recited in Claim 11, wherein the processing engine further includes:

a desktop manager, coupled to the component manager, wherein the desktop manager

collects attributes of the second software component; and

manipulates the second software component according to the attributes.

17. The apparatus as recited in Claim 16, wherein the desktop manager manipulates the second software component by causing the component manager to

download the second software component as part of executing an instance of the processing engine; and

replace an existing version of the second software component with the second software component that has been downloaded in the same instance.

18. A computer-readable medium carrying one or more sequences of instructions for managing a plurality of network devices on a network, which instructions, when executed by one or more processors, cause the one or more processors to:

detect a version change to a first software component out of the plurality of the software components; and

automatically identify a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component.

19. The computer-readable medium as recited in Claim 18, further comprising instructions which, when executed by the one or more processors, cause the one or more processors to automatically download the second software component.

20. The computer-readable medium as recited in Claim 19, further comprising instructions which, when executed by the one or more processors, cause the one or more processors to store a copy of the second software component in a cache.

21. The computer-readable medium as recited in Claim 20, further comprising instructions which, when executed by the one or more processors, cause the one or more processors to check version information of the second software component that is stored in the cache to determine whether to download the second software component.

22. The computer-readable medium as recited in Claim 18, further comprising instructions which, when executed by the one or more processors, cause the one or more processors to

collect attributes of the second software component; and

automatically manipulate the second software component according to the attributes.

23. The computer-readable medium as recited in Claim 22, further comprising instructions which, when executed by the one or more processors, cause the one or more processors to:

download the second software component; and

replace an existing version of the second software component with the second software component that has been downloaded.

24. An apparatus for managing versions of a plurality of software components on a network, comprising:

a user interface means; and

a processing means, coupled to the user interface, wherein the processing means further includes:

a detection means for detecting a version change to a first software component out of the plurality of the software components; and

a compatibility verification means for automatically identifying a second software component out of the plurality of the software components that needs to be changed to be compatible with the first software component, wherein the second software component depends on the first software component.

25. The apparatus as recited in claim 24, wherein the compatibility verification means automatically downloads the second software component.

26. The apparatus as recited in claim 24, wherein the compatibility verification means informs an operator of the apparatus of the second software component via the user interface means.

27. The apparatus as recited in Claim 25, wherein the compatibility verification means stores a copy of the second software component in a cache.

28. The apparatus as recited in Claim 27, wherein the compatibility verification means checks version information of the second software component that is stored in the cache to determine whether to download the second software component.

29. The apparatus as recited in Claim 24, wherein the processing means further includes:

a management means for

collecting attributes of the second software component; and

manipulating the second software component according to the attributes.

30. The apparatus as recited in Claim 29, wherein the management means further

downloads the second software component as part of executing an instance of the processing means; and

replaces an existing version of the second software component with the second software component that has been downloaded in the same instance.